Final Project Report

The restaurant landscape in New York City is very crowded, confusing, competitive for both business owners and customers. When deciding on a restaurant out of the plethora of options available no matter which neighborhood you are in, customer reviews play a deeply significant role in the final decision. According to a study from 2020, 90% of diners report researching reviews before picking a spot. These reviews build the reputation and brand image of anything being advertised online. Forbes magazine references a study revealing that 98% of customers "see reviews as an essential part of the decision-making process". In my personal experience, my friends and I always make sure that the customer rating of a restaurant is more than 4 stars on Google Reviews since we have endless options giving us no reason to pick anything below this standard,

PROBLEM:
The problem I am working to solve is figuring out what aspects or factors are involved in determining a restaurant's success in New York City. I have created a comprehensive and up-to-date dataset of restaurants in NYC and have used this information for various purposes, such as data analytics on multiple factors, sentiment analysis on reviews, neighborhood insights, operational insights, as well as pricing analysis. I was later able to use k-means clustering to visually represent high rated restaurants within each neighborhood as well.
Incorporates data integration, natural language processing (NLP), and visualization techniques discussed in lectures and papers, as well as a k-means clustering algorithm

STRATEGIC ASPECTS:
I have performed analysis on customer reviews, review counts, ratings, cuisine types, location, operating hours, price ranges, and NYC neighborhoods.

I had to use multiple API keys, from Google and Yelp to fetch restaurant data.
I had to organize and clean the data, such as labeling cuisine type for each restaurant based on the categories provided by the Yelp API or normalizing the hours if they were open 24 hours.
I had to fetch all restaurant and neighborhood details.

DATA USED:

- I chose 7 different neighborhoods within New York City and tried my best to have them be as spread apart as possible. The neighborhoods I chose were West Village, SoHo, Lower

East Side, Upper East Side, Midtown, Chinatown, Upper West Side. For these neighborhoods, I picked up 50 restaurants from each neighborhood.

- Why this dataset:
  - Different neighborhoods often specialize in specific cuisines, providing a rich dataset for analysis.
  - From high-end restaurants in the Upper East Side to affordable places in the Lower East Side, the dataset is able to capture a wide range of price points.
  - Different neighborhoods attract diverse demographics, impacting the types of restaurants and their popularity.
  - I selected 50 restaurants per neighborhood to ensure a reasonable sample size for analysis.

HOW I FETCHED DATA:

- API Selection:
  - I used the Google Places API and the Yelp Fusion API to fetch restaurant data.
    - These APIs provide comprehensive information about restaurants, including location, ratings, reviews, cuisine types, and more.
- Data Fetching
  - I wrote a Python script and the requests library to make API calls to Google Places and Yelp.

Google Places API: Fetch Place Details with Additional Fields

```python
import time
import requests

def get_google_places(api_key, location, radius, place_type):
    base_url = "https://maps.googleapis.com/maps/api/place/nearbysearch/json"
    results = []
    next_page_token = None

    while True:
        # API parameters
        params = {
            "key": api_key,
            "location": location,
            "radius": radius,
            "type": place_type,
        }
        if next_page_token:
            params["pagetoken"] = next_page_token

        # API request
        response = requests.get(base_url, params=params)
        response.raise_for_status()
        data = response.json()

        # Add results to the list
        results.extend(data.get("results", []))

        # Check for next_page_token
        next_page_token = data.get("next_page_token")
        if not next_page_token:
            break

        time.sleep(2)  # Wait for 2 seconds before the next request

    return results

# Google Places API Fetching Place Details
def get_google_place_details(api_key, place_id):
    url = f"{GOOGLE_BASE_URL}/place/details/json"
    params = {
        "key": api_key,
        "place_id": place_id,
        "fields": "name,formatted_address,rating,geometry,types,opening_hours,user_ratings_total,reviews"
    }
    response = requests.get(url, params=params)
    response.raise_for_status()
    return response.json().get("result", {})
```

- I specified the location and radius for each neighborhood to fetch relevant restaurant data.
    - I fetched neighborhood details based on a certain radius, since some neighborhoods are a larger size than other neighborhoods (can be seen in the image with API keys)
        - resulted in a few overlapping restaurants, but not enough to make a difference in our dataset
- I iterated through the API responses, extracting the necessary information for each restaurant.

Fetching Neighborhood Restaurant Data

```python
def fetch_neighborhood_restaurants(location, radius, place_type, google_api_key, yelp_api_key):
    google_places = get_google_places(google_api_key, location, radius, place_type)
    enriched_restaurants = []

    for place in google_places[:50]:  # Limit to 50 restaurants per neighborhood
        place_details = get_google_place_details(google_api_key, place["place_id"])
        google_name = place_details.get("name")
        google_address = place_details.get("formatted_address")
        google_lat = place_details["geometry"]["location"]["lat"]
        google_lng = place_details["geometry"]["location"]["lng"]

        # Additional fields from Google
        google_hours = clean_google_hours(place_details.get("opening_hours", {}).get("weekday_text", []))
        google_reviews = place_details.get("reviews", [])
        google_popularity = place_details.get("user_ratings_total", 0)
        google_photos = [photo.get("photo_reference") for photo in place_details.get("photos", [])]

        # Review text and ratings
        google_review_data = [
            {
                "text": review.get("text", ""),
                "rating": review.get("rating", None)
            }
            for review in google_reviews
        ]

        # Search Yelp for matches
        yelp_results = search_yelp_restaurant(yelp_api_key, location, google_name)
        yelp_match = None

        # Validate matches by comparing lat/lng
        for yelp_result in yelp_results:
            yelp_lat = yelp_result["coordinates"]["latitude"]
            yelp_lng = yelp_result["coordinates"]["longitude"]
            if is_match_by_location(google_lat, google_lng, yelp_lat, yelp_lng):
                yelp_match = yelp_result
                break

        # Additional fields from Yelp
        yelp_categories = [category["title"] for category in yelp_match.get("categories", [])] if yelp_match els
        yelp_transactions = yelp_match.get("transactions", []) if yelp_match else []
        yelp_menu_url = yelp_match.get("url", None) if yelp_match else None
        yelp_is_closed = yelp_match.get("is_closed", None) if yelp_match else None

        # Determine cuisine
        cuisine = label_cuisine(yelp_categories)
```

- Data cleaning and preprocessing
    - I cleaned the data to remove inconsistencies, errors, and irrelevant information.
    - I standardized the data format, ensuring consistency across different data sources.
    - I handled missing data by either imputing values or excluding incomplete records.

- Data Organization and Storage
    - I organized the extracted data into a structured format, such as a list of dictionaries.
    - I stored the data in a flattened JSON file for further analysis and processing.
- By doing this, I successfully used Google Places and Yelp APIs to fetch restaurant data for specific neighborhoods in New York City.

ADVANTAGES

- By combining Google Places and Yelp APIs, I was able to access a wide range of restaurant attributes, including location, ratings, reviews, cuisine types, price range, and hours of operation.
- By selecting diverse neighborhoods I ensured a representative sample of New York City's restaurant scene.
- Using APIs allowed me to tailor the data collection to specific needs, such as filtering by cuisine type, price range, or rating.
- This approach allows for scalability, since it can be easily scaled to include more neighborhoods or cities by modifying the API requests.

LIMITATIONS

- Although the Google Places API has a rate limit, it allowed me to have a sufficient amount to make as many API calls as I needed for the project. The real limitation was the Yelp API which only has a rate limit of 300 requests per day.
    - In order to combat this, I used multiple Yelp API keys. Since I only had a few API calls per day for the Yelp API, I created two separate yelp APIs. I split the API call into two different parts using the different API keys. I called the first API key for the first half of the data when fetching restaurant details, and switched to the second API key to fetch the details for the second half.

Fetching Restaurants Based on Location and Making API Calls

```python
# Define radius and place type
place_type = "restaurant"

# Neighborhoods and their lat/lng coordinates with dynamic radius based on location
# Rebalanced neighborhoods
NEIGHBORHOODS_CALL_1 = {
    "West Village": {"location": "40.7359,-74.0036", "radius": 1000},  # 1 km
    "SoHo": {"location": "40.7233,-74.0030", "radius": 1000},         # 1 km
    "Lower East Side": {"location": "40.7150,-73.9843", "radius": 1000},  # 1 km
}

NEIGHBORHOODS_CALL_2 = {
    "Upper East Side": {"location": "40.7736,-73.9566", "radius": 2000},  # 2 km
    "Midtown": {"location": "40.7549,-73.9840", "radius": 1500},      # 1.5 km
    "Chinatown": {"location": "40.7158,-73.9970", "radius": 1000},    # 1 km
    "Upper West Side": {"location": "40.7870,-73.9754", "radius": 2000},  # 2 km
}

YELP_API_KEY_1 = "AHw4onevVP1E2N1jdGv0xU8dMp2yY52XCxdNSVc2bhbotRfyJKvvemBXuEW3vjsyLx2A22idXD0fzYiiw5K7zkGNOSsYiI
YELP_API_KEY_2 = "B0WdruultLblnBmOMDL-faee_-BhaKu1oyC13_WLmz_0CU3FXB82EEG9LAF2FSvkZFwJSmUGLytCsTSlXSD1yrrTsSlHX4

try:
    # Fetch data using different API keys
    all_neighborhood_data_1 = fetch_all_neighborhoods(NEIGHBORHOODS_CALL_1, place_type, GOOGLE_API_KEY, YELP_API
    all_neighborhood_data_2 = fetch_all_neighborhoods(NEIGHBORHOODS_CALL_2, place_type, GOOGLE_API_KEY, YELP_API

    all_neighborhood_data = {**all_neighborhood_data_1, **all_neighborhood_data_2}

    # Save to a JSON file
    output_file = "nyc_restaurants_7x50.json"
    with open(output_file, "w") as f:
        json.dump(all_neighborhood_data, f, indent=4)

    print(f"Restaurant details for all neighborhoods saved to {output_file}")

except Exception as e:
    print(f"Critical error: {e}")

```

I used the AWS Relational Database Service (RDS) to host my PostgreSQL database. To efficiently manage and interact with the database, I used Postico, a database client, to establish a secure connection and execute my SQL queries.

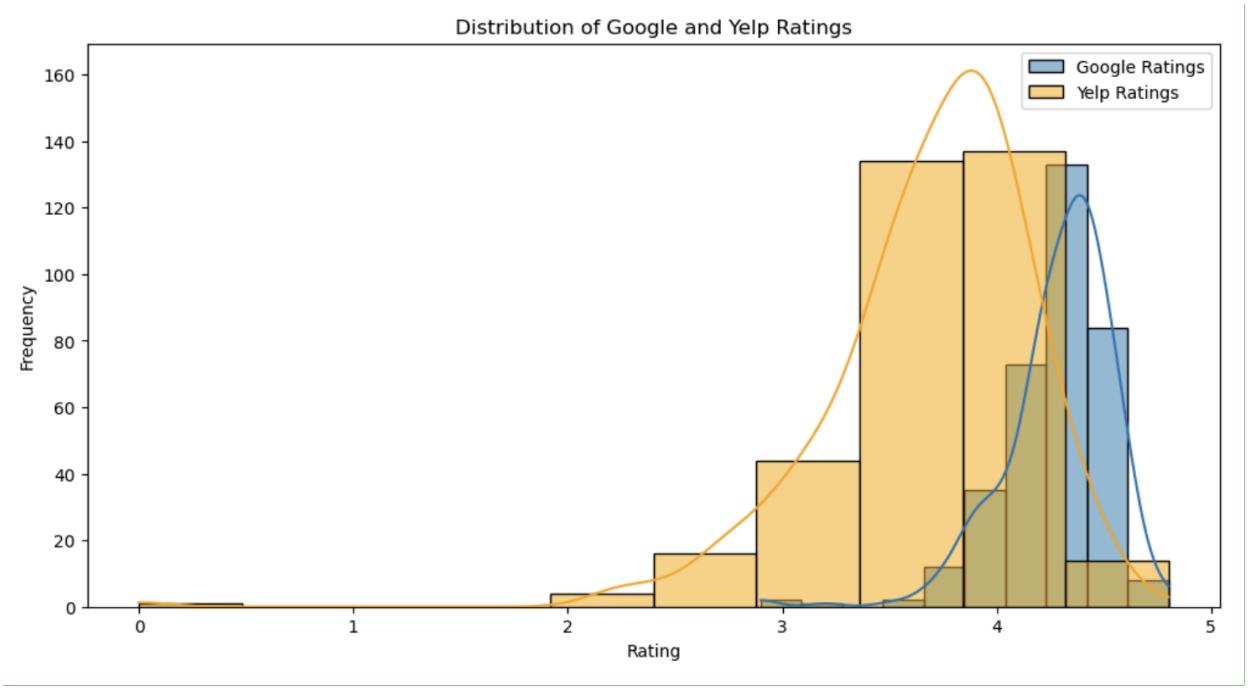
Next I had to create multiple different tables in order to keep everything standardized, uniform, and one-dimensional. So, I wrote multiple SQL queries to create tables in my database and later inserted all the data I fetched into the corresponding columns of the 5 tables I created.

```python
import psycopg2
from datetime import datetime

# Database connection details
DB_HOST = "restaurantdb.c3ka6a2sic3o.us-east-2.rds.amazonaws.com"
DB_PORT = 5432
DB_NAME = "postgres"
DB_USER = "aditisirohi"
DB_PASSWORD = "****"

# Connect to the PostgreSQL database
conn = psycopg2.connect(
    host=DB_HOST,
    port=DB_PORT,
    database=DB_NAME,
    user=DB_USER,
    password=DB_PASSWORD
)
cursor = conn.cursor()
```

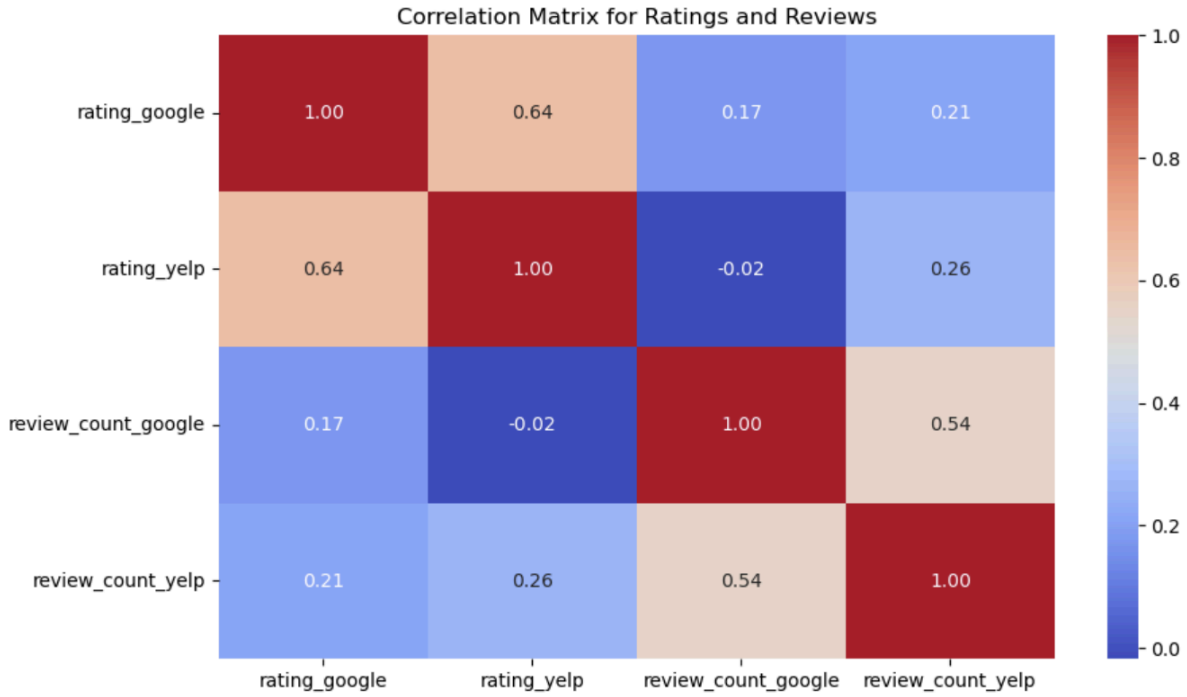
After the data was in the database, I was able to bring it into dataframes for clean data storage and for further analysis. Dataframes allowed me to use the functions for cleaning, sorting, merging, etc, and to explore specific subsets of data using filtering and indexing techniques. This simplified my data analysis process significantly.

I have been able to design experiments to see the role of cuisine type in restaurant success. I examined multiple cuisine types and analyzed if a certain type of cuisine does better in one neighborhood than another. I identified which cuisines had the highest number of reviews and which had the lowest. I identified whether certain cuisines had higher ratings than others and analyzed whether location plays a significant role in this. I also analyzed whether there was a correlation between a restaurant or cuisine having a higher number of reviews and a higher rating.
Next, I was able to perform sentiment analysis and finally develop a K-means clustering algorithm

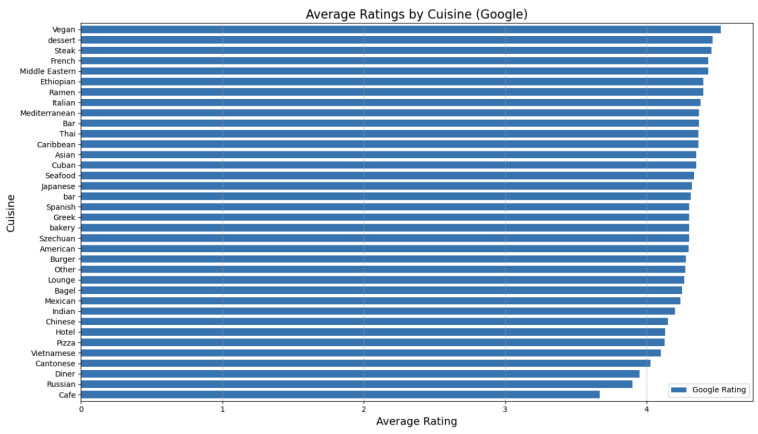
KEY FINDINGS
Starting off, I found that most of the restaurant ratings were distributed between 3.5 and 4.5 across both google and yelp ratings

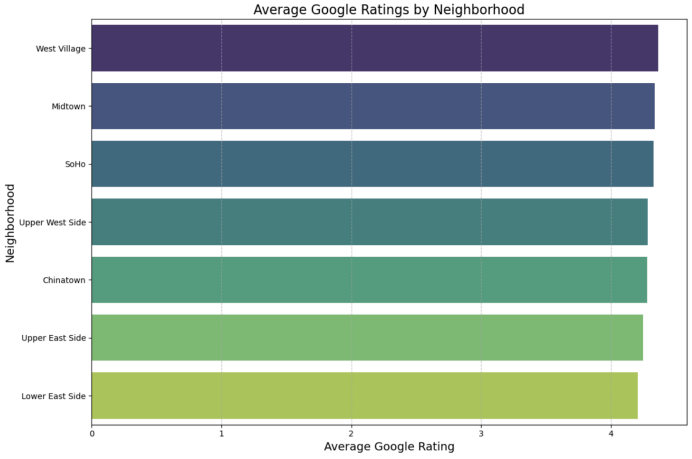


According to the correlation matrix, a higher review count does not correlate directly with a high rating for a restaurant across both yelp and google. However, there is a high correlation between google rating and yelp rating, so the restaurants are similarly rated on both platforms. The correlation between the google review count and yelp review count is also somewhat high, so the review counts for a restaurant on both platforms are also similar.

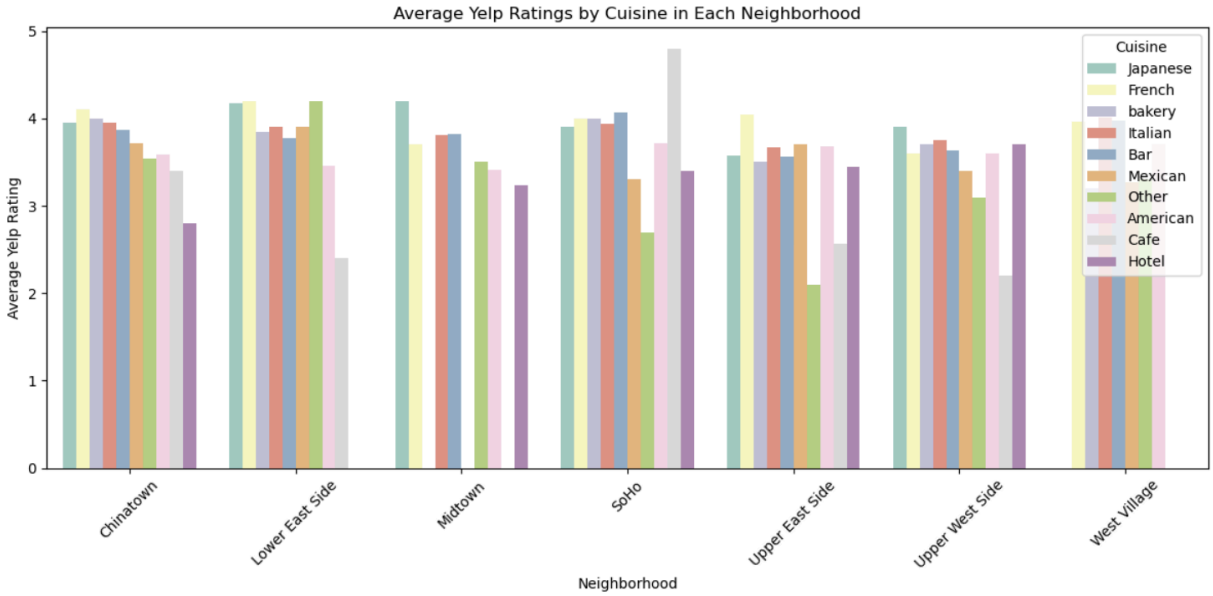


Some cuisines tend to receive higher average ratings than others and it varies by each neighborhood. So the neighborhood the restaurant is in can highly affect its success.



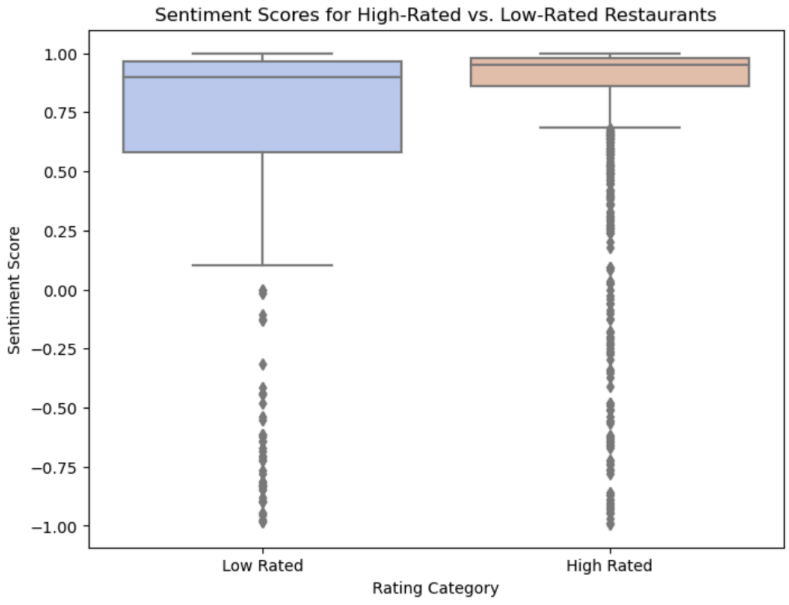


I also designed an experiment where I tested whether neighborhood affects rating for a specific cuisine. Turns out, it does!

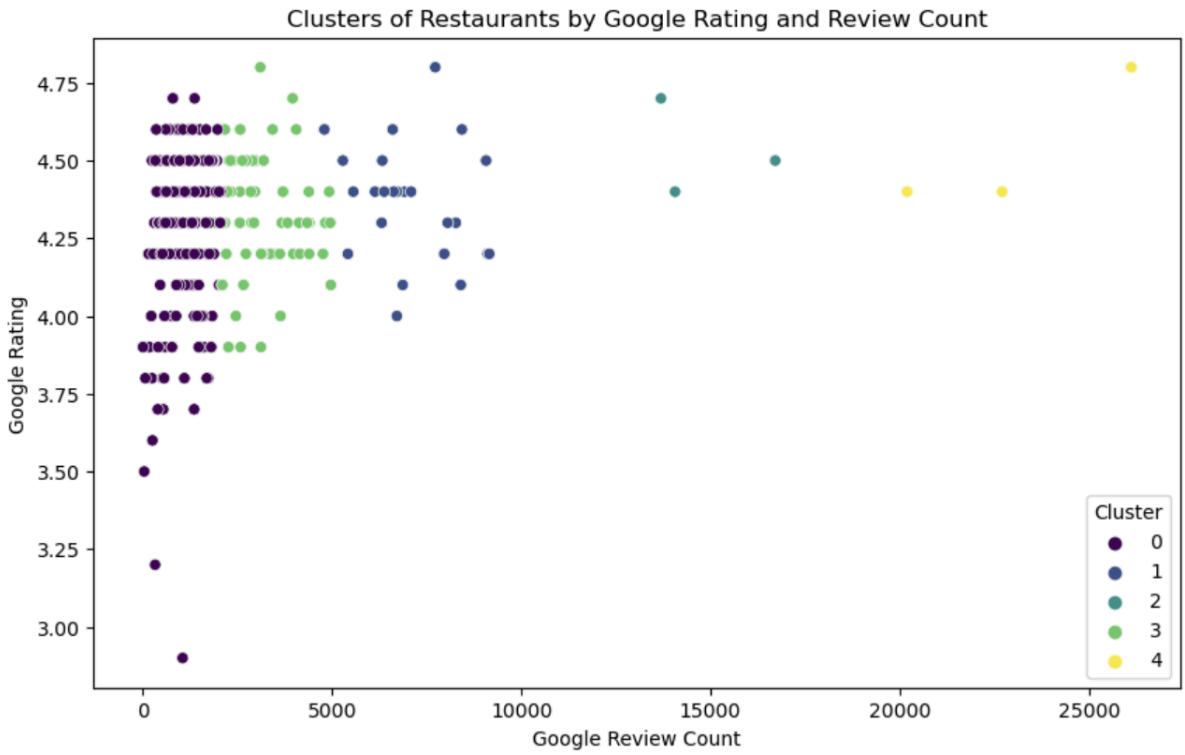


For example, in SoHo, cafes rated significantly higher than any other cuisine. In Lower East Side, Japanese and French restaurants tend to gain more popularity. In West Village, Italian and in Midtown, French restaurants again.

I performed sentiment analysis where I saw a trend between sentiment score and rating. The low rated restaurants seems to have reviews that have lower sentiment scores(more negative) and the high rated restaurants seems to have reviews that have higher sentiment scores(more positive), which is just as expected.



Lastly I performed K-means clustering on my dataset.



- Cluster 0 (Purple): Concentrated at the lower review count and moderately low ratings (3.5 to 4.0). Likely represents underperforming or less popular restaurants.
- Cluster 1 (Blue): Spread across medium to high review counts and moderately high ratings (4.0+). Likely represents well-rated and somewhat popular restaurants.
- Cluster 2 (Green): Represents restaurants with lower review counts but relatively higher ratings (4.2+). These could be newer or niche restaurants that cater to a specific audience.
- Cluster 3 (Teal): High review counts (10,000 to 15,000) and consistent high ratings (4.2+). Likely very popular restaurants with a strong reputation.
- Cluster 4 (Yellow): Outliers with extremely high review counts (20,000+) and top-tier ratings (~4.7+). These are likely iconic or flagship restaurants.

GITHUB LINK

https://github.com/aditisirohi/cs210-final-project.git